End-Semester Exam - Deep Learning (DL) [Winter 2025]

20 Marks [120 Minutes]

Instruction

- ☐ Must be filled completely (■) for any answer to be valid, partially filled or tick mark will not be valid.
- Beginner questions carry -ve marking of 0.5.

- **Beginner** **0.5 Marks Each**

1. Given the loss function: $\mathcal{L}(\theta) = \mathbb{E}[\ell(y, f_\theta(x))] + \lambda\|\theta\|_2^2$, which probabilistic prior corresponds to this regularization?
   (a) ☐ Laplace (b) ☐ Gaussian (c) ☐ Cauchy (d) ☐ Dirichlet

   B

2. For a 2D convolution with kernel `[[a, b], [c, d]]` applied to an input `[[., .], [., .]]` what's the value of $\frac{\partial Y}{\partial a}$ at position (1, 1) using 'valid' padding?
   (a) ☐ 1 (b) ☐ 2 (c) ☐ 3 (d) ☐ 4

   A

3. The PAC-bound for empirical risk $R_{emp}(h) \leq R_{true}(h) + \mathcal{O}\left(\sqrt{\frac{VC(H)}{n}}\right)$ implies:
   (a) ☐ Larger models always generalize better (c) ☐ Sample complexity grows quadratically with VC-dim
   (b) ☐ Model complexity must scale with $\sqrt{n}$ (d) ☐ Regularization inversely relates to $\sqrt{n}$

   C

4. Which condition violates the Universal Approximation Theorem for MLPs?
   (a) ☐ Using ReLU activations (c) ☐ Linear output layer
   (b) ☐ Bounded depth with unbounded width (d) ☐ Discrete-valued hidden units

   D

5. The Perceptron update $\Delta w = \eta(y - \hat{y})x$ corresponds to minimizing:
   (a) ☐ Hinge Loss (b) ☐ 0-1 Loss (c) ☐ Cross-entropy (d) ☐ Squared error

   B

6. For loss $\mathcal{L} = (\sigma(wx + b) - y)^2$ where σ is sigmoid, compute $\frac{\partial \mathcal{L}}{\partial w}$:
   (a) ☐ $2(\sigma - y)\sigma(1 - \sigma)x$ (c) ☐ $(\sigma - y)(1 - \sigma^2)x$
   (b) ☐ $2(\sigma - y)x$ (d) ☐ $2(\sigma - y)(1 - \sigma)$

   A

7. A 3×3 convolution with dilation=2 over a 7×7 input produces output size:
   (a) ☐ 3×3 (b) ☐ 5×5 (c) ☐ 7×7 (d) ☐ 9×9

   A

8. The number of parameters in a depthwise separable convolution with $C_{in} = 32$, $C_{out} = 64$, kernel=3×3 is:
   (a) ☐ 2336 (b) ☐ 1152 (c) ☐ 896 (d) ☐ 448

   A

9. In ResNet's residual block $F(x) + x$, the gradient $\frac{\partial \mathcal{L}}{\partial x}$ equals:
   (a) ☐ $\frac{\partial \mathcal{L}}{\partial F(x)}$ (c) ☐ $\frac{\partial \mathcal{L}}{\partial F(x)} \cdot (1 + \frac{\partial F}{\partial x})$
   (b) ☐ $\frac{\partial \mathcal{L}}{\partial F(x)} + \frac{\partial \mathcal{L}}{\partial x}$ (d) ☐ $\frac{\partial \mathcal{L}}{\partial F(x)} + \frac{\partial \mathcal{L}}{\partial (F+x)}$

   B

10. Minimizing KL divergence $KL(q\|p)$ where q is posterior and p is prior corresponds to:
    (a) ☐ Maximum likelihood (c) ☐ MAP estimation
    (b) ☐ Variational inference (d) ☐ Empirical Bayes

    B

11. Suppose you have examples $(\mathbf{x}_i, y_i)$ in $\mathbb{R}^d$ separable by a unit-length vector w^* with margin $\gamma = \min_i [y_i (w^* \cdot x_i)]$ and let $R = \max_i \|x_i\|$. Running the Margin Perceptron on this data makes at most how many mistakes?

(a) ☐ $\left(\frac{R}{\gamma}\right)^2$ (b) ☐ $4\left(\frac{R}{\gamma}\right)^2$ (c) ☐ $8\left(\frac{R}{\gamma}\right)^2 + 4\left(\frac{R}{\gamma}\right)$ (d) ☐ $2\left(\frac{R}{\gamma}\right)^2 + 4\left(\frac{R}{\gamma}\right)$

C

12. In 1D convolution, what is the primary effect of increasing the dilation rate d (with kernel size k fixed), while keeping stride = 1 and no padding?

(a) ☐ It increases the number of parameters in the model
(b) ☐ It increases the receptive field
(c) ☐ It reduces the receptive field size
(d) ☐ It increases the activation map size

C

13. What is a key trade-off introduced by group convolution in deep neural networks?

(a) ☐ Overlapping filter banks across groups induce implicit parameter tying across unrelated features
(b) ☐ Increased non-linearity at the cost of convergence speed
(c) ☐ Stronger spatial invariance due to disconnected receptive fields
(d) ☐ Reduced parameter sharing across channels, potentially limiting cross-channel feature learning

D

14. A model using BatchNorm performs well during training but poorly at inference. What is the most plausible explanation?

(a) ☐ BatchNorm disables learnable parameters (scale and shift) in eval mode
(b) ☐ BatchNorm uses stale running statistics that don't reflect test data distribution
(c) ☐ Gradients through BatchNorm layers are blocked in evaluation mode
(d) ☐ Inference reuses the last mini-batch's batch statistics, which may be noisy

B

15. Which of the following correctly describes the gradient clipping by norm technique?

(a) ☐ $\hat{g} \leftarrow \hat{g}^2 +$ threshold if $\|\hat{g}\| \geq$ threshold
(b) ☐ $\hat{g} \leftarrow c$ if $\|\hat{g}\| \geq$ threshold, where c is a hyperparameter
(c) ☐ $\hat{g} \leftarrow$ threshold $\cdot \frac{\hat{g}}{\|\hat{g}\|}$ if $\|\hat{g}\| \geq$ threshold
(d) ☐ $\hat{g} \leftarrow \text{ReLU}(\hat{g})$ to remove negative gradients

C

16. In a deep neural network with L layers subject to per-tensor quantization with bit-width b, the mutual information between the input X and output Y can be bounded according to the data processing inequality with quantization noise. If we denote the mutual information without quantization as $I(X;Y)$ and with quantization as $I_Q(X;Y)$, then:

(a) ☐ $I_Q(X;Y) \leq I(X;Y) - \sum_{l=1}^{L} H(W_l)$, where $H(W_l)$ is the entropy of quantization noise in layer l
(b) ☐ $I_Q(X;Y) \leq I(X;Y) - \sum_{l=1}^{L} D_{KL}(P(Z_l) \| p_Q(Z_l))$, where D_{KL} is the Kullback-Leibler divergence between original and quantized activation distributions
(c) ☐ $I_Q(X;Y) \geq I(X;Y) - \sum_{l=1}^{L} \frac{\sigma_{w_l}^2 \Delta_l^2}{12}$, where $\sigma_{w_l}^2$ is the variance of weights in layer l and Δ_l is the quantization step size
(d) ☐ $I_Q(X;Y) \geq I(X;Y) - \sum_{l=1}^{L} \log_2(1 + \frac{\sigma_{z_l}^2}{2^{2b}\sigma_{n_l}^2})$, where $\sigma_{z_l}^2$ is the variance of activations and $\sigma_{n_l}^2$ is the variance of quantization noise

17. Apply transpose convolution (stride = 2, padding = 1) to input $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ using filter $\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$. Let X = output[0,0], Y = output[2,0], Z = output[2,2]. What are their values?

(a) ☐ $X = -1, Y = -4, Z = -2$
(b) ☐ $X = 0, Y = -6, Z = -4$
(c) ☐ $X = 2, Y = -6, Z = -4$
(d) ☐ $X = 1, Y = -2, Z = 0$

A

18. In Variational Autoencoders (VAEs), which identity correctly relates the marginal log-likelihood $\log p(x)$ to a variational distribution $q(z|x)$?

(a) ☐ $\log p(x) = \mathbb{E}_{q(z|x)}[\log p(x|z)] + \text{KL}(q(z|x) \| p(z))$
(b) ☐ $\log p(x) = \mathbb{E}_{q(z|x)}[\log p(x, z)] - \mathbb{E}_{q(z|x)}[\log q(z|x)]$
(c) ☐ $\log p(x) = \log \mathbb{E}_{q(z|x)}\left[\frac{p(x,z)}{q(z|x)}\right]$
(d) ☐ $\log p(x) = \text{KL}(q(z|x) \| p(z|x)) + \mathbb{E}_{q(z|x)}[\log p(x|z)] - \text{KL}(q(z|x) \| p(z))$

D

19. What is the primary purpose of the reparameterization trick in VAEs?

(a) ☐ To reduce computational complexity of the VAE (c) ☐ To enable backpropagation through stochastic nodes

(b) ☐ To minimize the reconstruction error (d) ☐ To normalize input data

B

20. Given

$$X = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}, \quad W = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}.$$

Forward pass is: $X \to 2\times2$ valid conv $\to$ ReLU $\to 2\times2$ max-pool $\to$ scalar P. Backward pass via DeconvNet is: $P \to$ unpool (1 at pooled-max) $\to$ ReLU backward (zero out negative grads) $\to$ full transposed conv with $W \to$ saliency map. If we apply the DeconvNet to P, which output will be produced?

(a) ☐ $\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}$ (c) ☐ $\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$

(b) ☐ $\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 2 \\ 0 & 2 & 2 \end{bmatrix}$ (d) ☐ $\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

A

21. For a GAN, let $p = D(G(z))$ be the discriminator's output on a generated sample. Which generator objective gives a larger gradient magnitude when p is near 0?

(a) ☐ $L_G^{\text{mm}} = \log(1 - p)$ (b) ☐ $L_G^{\text{ns}} = -\log(p)$ (c) ☐ $L_G^{\text{LSGAN}} = \big(D(G(z)) - 1\big)^2$

(d) ☐ $L_G^{\text{W}} = -D(G(z))$

B. Explanation: For the minimax loss $L_G^{\text{mm}} = \log(1 - p)$,

$$\frac{d}{dp}\log(1 - p) = -\frac{1}{1 - p},$$

whose magnitude remains bounded (1) as $p \to 0$. By contrast, for the non-saturating loss $L_G^{\text{ns}} = -\log(p)$,

$$\frac{d}{dp}\big[-\log(p)\big] = \frac{1}{p},$$

which $\to\infty$ when $p \to 0$. Thus the non-saturating objective (B) uniquely avoids vanishing gradients early in training.

22. In an LSTM cell with update rule

$$C_t = f_t C_{t-1} + i_t \tilde{C}_t,$$

suppose the forget- and input-gate values are held constant at $f_t = f$ and $i_t = i$ for all t. After T steps, the contribution of the initial state C_0 to C_T is f^T. Which pair (f, i) gives strong long-term memory (i.e. $f^T \approx 1$ for large T) while still allowing new information to enter (i.e. $i > 0$)?

(a) ☐ $f = 1.0,\ i = 0.0$ (b) ☐ $f = 0.99,\ i = 0.01$ (c) ☐ $f = 0.9,\ i = 0.9$ (d) ☐ $f = 0.5,\ i = 0.5$

Answer: (B) $f = 0.99,\ i = 0.01$.

(a) Option A ($f = 1, i = 0$) perfectly preserves C_0 ($f^T = 1$) but admits no new information ($i = 0$), so it cannot learn anything beyond the initial state.

(b) Option C ($f = 0.9, i = 0.9$) allows plenty of updating (i large) but exponentially decays past memory ($0.9^T \to 0$ as T grows).

(c) Option D ($f = 0.5, i = 0.5$) suffers severe forgetting (0.5^T) and over-writes old content rapidly.

(d) Option B balances the two: with $f = 0.99$, $0.99^T \approx e^{-0.01T}$ stays near 1 for moderate T, while $i = 0.01$ still injects new information—thus preserving long-range gradients yet allowing learning of fresh inputs.

23. In attention mechanisms and other models using masked softmax, certain positions (e.g. padding tokens or future tokens) are "masked" by replacing their logits with a large negative value (e.g. -10^6). What is the primary purpose of this masking before applying the softmax operation?

(a) ☐ To amplify masked positions so they dominate the distribution

(b) ☐ To speed up the softmax computation by reducing dynamic range

(c) ☐ To avoid division by zero during normalization

(d) ☐ To ensure masked positions receive near-zero probability after softmax

D. Explanation: Masked elements are set to a large negative value so that after applying softmax, their exponentials become near-zero, effectively removing their influence from the distribution.

24. Given tensors A of shape (b, n, m) and B of shape (b, m, p), which PyTorch operation efficiently computes the batched matrix products $\{A_i B_i\}$ for $i = 1, \ldots, b$?

(a) ☐ `torch.matmul(A, B)` (b) ☐ `torch.bmm(A, B)` (c) ☐ `A @ B` using Python loops

(d) ☐ `torch.einsum('bnm,bmp->bnp', A, B)`

B. Explanation: `torch.bmm` is optimized specifically for batch matrix multiplication of 3D tensors. While `torch.matmul` can also perform this, `bmm` is more direct and efficient in this specific case.

25. What is the time complexity of a full self-attention layer on a sequence of length l with embedding dimension h?

(a) ☐ $\mathcal{O}(l h^2 + l^2 h)$ (b) ☐ $\mathcal{O}(l h + l^2)$ (c) ☐ $\mathcal{O}(l^2 h + l h^2)$ (c) ☐ $\mathcal{O}(l^3)$

C. Explanation: Self-attention involves computing attention weights ($l^2 h$) and then applying them to the values, plus linear projections ($l h^2$). Hence the total complexity is $\mathcal{O}(l^2 h + l h^2)$.

26. What is the effect of mixed-precision training using `torch.cuda.amp`?

(a) ☐ Reduces memory usage and speeds up computation by using FP16 where possible

(b) ☐ Increases model robustness by adding noise to gradients

(c) ☐ Increases numerical precision by enforcing FP64 operations

(d) ☐ Forces all computations to use FP32

A. Explanation: Mixed-precision training with `torch.cuda.amp` uses FP16 where it's safe to do so, reducing memory usage and improving speed, while keeping certain operations in FP32 to maintain stability.

- **Intermediate** **2 Marks Each**

1. Using a simple example in $\mathbb{R}^1$, explain mathematically why Wasserstein- distance provides meaningful gradients for GAN training when real and generated distributions have disjoint support, while Jensen-Shannon divergence fails to do so.
The key difference between the Wasserstein-1 distance and Jensen-Shannon (JS) divergence is how they behave when two distributions have non-overlapping supports. Suppose P and Q are two distributions such that $\text{supp}(P) \cap \text{supp}(Q) = \emptyset$. The JS divergence is defined as:

$$JS(P\|Q) = \frac{1}{2} D_{KL}(P\|M) + \frac{1}{2} D_{KL}(Q\|M), \quad \text{where} \quad M = \frac{1}{2}(P + Q)$$

Since P and Q never overlap, M simply puts half its mass on each distribution separately. Then:

$$D_{KL}(P\|M) = D_{KL}(Q\|M) = \log 2$$

Thus:

$$JS(P\|Q) = \log 2 \approx 0.693$$

The value is constant and independent of how far apart P and Q are. Hence, the gradient $\nabla_\theta JS(P\|Q_\theta) = 0$, when supports are disjoint. There is no useful gradient for learning. Whereas the Wasserstein distance is defined as:

$$W_1(P, Q) = \inf_{\gamma \in \Pi(P,Q)} \mathbb{E}_{(x,y)\sim\gamma}[|x - y|]$$

where $\Pi(P, Q)$ is the set of all couplings of P and Q. Even when supports are disjoint, $W_1(P, Q)$ measures how far mass must be transported, and it varies smoothly with the distance between P and Q. Thus, the gradient is nonzero and provides a meaningful direction for optimization. Simple Example in $\mathbb{R}^1$, take:

$$P = \delta_0 \quad \text{(Dirac delta at } x = 0\text{)}$$

$$Q_\theta = \delta_\theta \quad \text{(Dirac delta at } x = \theta\text{)}$$

JS Divergence:

- For any $\theta \neq 0$, $JS(P\|Q_\theta) = \log 2$.
- Thus, $\nabla_\theta JS(P\|Q_\theta) = 0$.

Wasserstein Distance:

- $W_1(P, Q_\theta) = |\theta|$,
- Thus, $\nabla_\theta W_1(P, Q_\theta) = \text{sign}(\theta)$.

The Wasserstein gradient points in the correct direction to bring Q_θ toward P. Thus, Wasserstein distance makes GAN training much more stable, especially in early stages.

2. In generative sequence modeling, the autoregressive property is crucial for properly modeling conditional distributions. Specifically, demonstrate why causal convolutions allow us to model $p(x_t|x_{<t})$ directly without contamination from future information, and prove that this is equivalent to enforcing the conditional independence: $p(x_t|x_{<t}) = p(x_t|x_{<t}, x_{>t})$ when using causal convolutions.
In autoregressive sequence modeling, we want to model the joint probability of a sequence $x = (x_1, x_2, \ldots, x_T)$ by factorizing it according to the chain rule of probability:

$$p(x_1, x_2, \ldots, x_T) = \prod_{t=1}^{T} p(x_t|x_{<t})$$

where $x_{<t} = (x_1, x_2, \ldots, x_{t-1})$ represents all elements in the sequence before time t. For this factorization to be valid for generative modeling, each conditional distribution $p(x_t|x_{<t})$ must depend only on past elements and not on future elements $x_{>t} = (x_{t+1}, \ldots, x_T)$.
Proof:
In a causal convolutional network, the output $y[t]$ is functionally independent of all inputs $x[t+m]$ for $m > 0$. Use induction on the layer depth l. For the first layer, the output at time t is:

$$h^{(1)}[t] = \sigma\left(\sum_{i=0}^{k-1} w^{(1)}[i] \cdot x[t-i]\right)$$

The furthest "look-back" is to position $t - (k-1)$. There is no dependency on any position $t+m$ where $m > 0$. Assume that for layer l, $h^{(l)}[t]$ depends only on inputs up to time t (i.e., $x[t], x[t-1], \ldots$) (from above). For layer $l+1$:

$$h^{(l+1)}[t] = \sigma\left(\sum_{i=0}^{k-1} w^{(l+1)}[i] \cdot h^{(l)}[t-i]\right)$$

each $h^{(l)}[t-i]$ depends only on inputs up to time $t-i$. Therefore, $h^{(l+1)}[t]$ can only depend on inputs up to time t, and not on any future input $x[t+m]$ where $m > 0$. By the principle of induction, this holds for all layers in the network.

- **Advance** **3 Marks Each**

1. Consider a neural network with parameters $\theta \in \mathbb{R}^d$ and a pruning operation $P(\theta, m)$ where $m \in \{0,1\}^d$ is a binary mask. Using the Fisher Information Matrix $F(\theta) = \mathbb{E}[\nabla \ell(\theta) \nabla \ell(\theta)^T]$, where $\ell(\theta)$ is the loss function, derive the change in expected loss when pruning parameters according to mask m:

$$\Delta L(\theta, m) \approx \frac{1}{2} \sum_{i:m_i=0} \frac{\theta_i^2 F_{ii}}{\sum_{j=1}^{d} \theta_j^2 F_{jj}} \cdot \|\theta\|_F^2$$

where $\|\theta\|_F^2$ represents the Fisher-weighted norm of parameters. (Hint: Taylor Expansion, Fisher Information Matrix)
When we prune a neural network, we use a binary mask $m \in \{0,1\}^d$ to set certain parameters to zero:

- $m_i = 0$ means parameter θ_i is pruned (set to zero)
- $m_i = 1$ means parameter θ_i is kept

The pruned parameter vector is $\theta \odot m$, where $\odot$ is element-wise multiplication. The change in loss after pruning is:

$$\Delta L(\theta, m) = L(\theta \odot m) - L(\theta)$$

We can approximate this change using a second-order Taylor expansion:

$$L(\theta \odot m) \approx L(\theta) + \nabla L(\theta)^T (\theta \odot m - \theta) + \frac{1}{2} (\theta \odot m - \theta)^T H(\theta) (\theta \odot m - \theta)$$

where $H(\theta)$ is the Hessian matrix of the loss function. Since $\theta \odot m - \theta = -\theta \odot (1 - m)$, we can rewrite:

$$\Delta L(\theta, m) \approx -\nabla L(\theta)^T (\theta \odot (1 - m)) + \frac{1}{2} (\theta \odot (1 - m))^T H(\theta) (\theta \odot (1 - m))$$

For a well-trained model, $\nabla L(\theta) \approx 0$ because the model is at or near a local minimum. Thus, the first-order term vanishes:

$$\Delta L(\theta, m) \approx \frac{1}{2} (\theta \odot (1 - m))^T H(\theta) (\theta \odot (1 - m))$$

The Hessian matrix $\mathbf{H}(\theta)$ can be approximated using the Fisher Information Matrix (FIM):

$$F(\theta) = \mathbb{E}[\nabla \ell(\theta) \nabla \ell(\theta)^T]$$

where the expectation is taken over the data distribution. For computational simplicity, use a diagonal approximation of the FIM:

$$F(\theta) \approx \text{diag}(F_{11}, F_{22}, \ldots, F_{dd})$$

With this approximation and recalling that $(\theta_i - m_i)^2 = (\theta_i - m_i)$ for binary m_i:

$$\Delta L(\theta, m) \approx \frac{1}{2} \sum_{i=1}^{d} (\theta_i - m_i) \theta_i^2 F_{ii} = \frac{1}{2} \sum_{i:m_i=0} \theta_i^2 F_{ii}$$

Express this as a relative change by normalizing with the total Fisher-weighted norm:

$$\Delta L(\theta, m) \approx \frac{1}{2} \sum_{i:m_i=0} \frac{\theta_i^2 F_{ii}}{\sum_{j=1}^{d} \theta_j^2 F_{jj}} \cdot \|\theta\|_F^2$$

where $\|\theta\|_F^2 = \sum_{j=1}^{d} \theta_j^2 F_{jj}$ is the Fisher-weighted norm of the parameters.

- $\theta_i^2 F_{ii}$ represents the importance of parameter θ_i
- Parameters with smaller values of $\theta_i^2 F_{ii}$ contribute less to loss when pruned
- For optimal pruning, remove parameters with the smallest values of $\theta_i^2 F_{ii}$

The formula quantifies exactly how much loss is expected to incur when removing specific parameters, allowing to make informed decisions about which parameters to prune while minimizing performance degradation.